Serial No. 08/880,616
Art Unit No. 2126

REMARKS

Claims 1-14 and 16-20 were pending in the patent application. By this amendment, Applicants amend Claims 1, 4, 8-9, 11, 13, and 18 and cancel Claims 6-7 and 10.

The Examiner has rejected Claims 1-5 under 35 USC 103(a) as being unpatentable over Zolnowsky; and Claims 6-13 and 16-20 under 35 USC 103 as being unpatentable over the teachings of Zolnowsky in view of Cameron. For the reasons set forth below, Applicants respectfully assert that all of the remaining pending claims, as amended, are patentable over the cited prior art.

The present invention provides a scheduling system and method for a multinode UNIX-based environment. Under the invention, at least one local scheduler prioritizes processes in accordance with a global prioritized schedule which is generated at the global scheduler. The local scheduler maintains a local priority list of ready-to-execute tasks correlated with local processes, which list is updated in accordance with the global prioritized schedule provided from the global scheduler. As set forth in the independent claims, the present invention

provides a method and system for performing the steps in a UNIX-based environment of scheduling a plurality of tasks of more than one application among processes on at least one computing node, in a system having a global scheduler and more than one computing node having a local scheduler and a plurality of local processes, comprising the steps of: dynamically creating and updating a global prioritized schedule of a plurality of tasks based on application information and local process information, said schedule including tasks of the more than one application; communicating the global prioritized schedule to the more than one computing nodes; and, at the local computing nodes, determining correspondence between the plurality of tasks and the plurality of local processes and dynamically prioritizing local processes in a local priority list in accordance with the global prioritized schedule to allow simultaneous execution of tasks from the more than one application.

The present approach of global and local scheduling minimizes unused CPU time which would otherwise occur when an individual task would be temporarily blocked or suspended waiting for I/O. The local scheduling takes into

consideration not just one task having a global priority rating but all of the tasks on a global prioritized schedule. In addition, the local scheduler does not simply look at one task, that task being the one having highest priority. Rather, the local scheduler creates a new local priority list (or schedule) of a plurality of tasks based on the global prioritized schedule information. Moreover, the local computing node communicates local process information (including such things as process availability and task completion information) to the global scheduler, thereby triggering generation of a new updated global prioritized schedule. The global scheduler continually, dynamically updates its global prioritized schedule based on received application information and local process information, and then communicates each dynamically updated global prioritized schedule to the more than one computing nodes. As such, the present invention is able to handle optimized parallel processing of multiple applications on multiple processes at more than one node without encountering deadlock situations caused by either tasks awaiting execution or contention.

Applicants respectfully assert that the Zolnowsky patent does not teach or suggest the invention as claimed. The Zolnowsky patent provides a dispatcher model which maintains a global dispatch queue for non-bound higher priority real time threads and provides a local dispatch queue and scheduler for each processor (Fig 5). Each processor in the multiprocessor environment, when it is available to execute a task, checks the global real-time queue and selects the highest priority candidate thread in the global real-time queue, unless there is a higher priority candidate thread in its own dispatch queue, and selects the highest priority thread for execution. The processor may additionally select a thread from a dispatch queue of another processor (Col. 8, lines 30-36). When a processor selects a thread from the global real-time queue, it broadcasts to other processors that it has selected the item to avoid duplication/contention. Even after thread selection but before execution, the processor rechecks the global real-time queue to ensure that no higher priority thread had been placed there. If a higher priority thread has appeared on the global real-time queue, the processor places the selected thread back on "some queue" (Col. 9,

line 47) and starts the process again. The Zolnowsky patent teaches that each processor's scheduler look at one thread at the global real-time queue at a time as compared to one thread at its own queue. The Zolnowsky patent does not teach or suggest that a global scheduler create a global prioritized schedule of more than one task and communicate that schedule of a plurality of tasks to local computing nodes at which a local scheduler updates a local prioritized schedule including more than one task.

Under the Zolnowsky patent, there is one scheduler for each one processor, each of which has an associated dispatch queue. Zolnowsky neither teaches nor suggests that computing nodes have more than one processor, defined as more than one "process" throughout the present application (see: e.g., page 1, line 13-page 2, line 5; page 6, lines 6 and 15). Rather, as clearly illustrated in Fig. 5 of Zolnowsky, there is one scheduler per processor, which one scheduler takes either the next thread from the local dispatch queue or the next thread from the global real-time queue. Applicants respectfully assert that Zolnowsky neither teaches nor suggests a multi-processor/multi-process environment wherein one local scheduler is associated with

one computing node, wherein each node has a plurality of local processes, as is explicitly claimed.

Applicants further contend that the Zolnowsky patent does not teach or suggest the creation or use of a global prioritized schedule. Zolnowsky provides a global real-time queue from which only one highest priority task at a time can be seen and removed. Applicants contend that providing a global real-time queue from which one task at a time can be viewed and removed is not the same as or suggestive of creating, updating, and communicating a global prioritized schedule of a plurality of tasks. Under Zolnowsky, a process can only see the priority level of one thread from the global real-time queue at a time and can remove that one thread. Zolnowsky's processor cannot see a schedule of a plurality of "real-time" tasks. Moreover, Zolnowsky does not communicate its real-time queue to a local node or processor. The processor must access that information from the global real-time queue (501 of Fig. 5). It is also to be noted that Zolnowsky allows a processor to place a selected thread on any queue if a higher priority thread is detected at the real-time queue. As such, a thread with a high global priority can be placed on a queue other than the

global real-time queue. Clearly such an arrangement cannot be said to obviate the creation, updating and communication of a dynamically maintained global prioritized schedule of a plurality of tasks.

Applicants further contend that the Zolnowsky patent does not teach or suggest the claimed at least one local scheduler associated with each of more than one computing nodes. While Zolnowsky provides one scheduler per processor, that scheduler does not include means for receiving a global prioritized schedule, means for ascertaining which of a plurality of tasks are assigned tasks, being assigned to each of the plurality of local processes, means for prioritizing the assigned processes, and means to update a local priority list to include the assigned processes in accordance with said global prioritized schedule, as is taught and claimed by the present application. The Zolnowsky schedulers (i.e., the one scheduler per processor) do not adhere to a prioritized schedule obtained from a global scheduler, but rather "...determine when and which threads are to be dispatched for execution on the system processors" (see: column 7, lines 15-20) in accordance with the process flow illustrated in Zolnowsky's Figure 7. Clearly the

Serial No. 08/880,616
Art Unit No. 2126

Zolnowsky patent does not provide any teaching or suggestion of the claimed system or method wherein both global and local schedulers are available and wherein a global prioritized schedule of a plurality of tasks is created, updated, and communicated to the local schedulers for updating a local prioritized list of a plurality of tasks. Accordingly, Applicants believe that Claims 1-5 are not rendered obvious by the Zolnowsky patent.

The Examiner has rejected the remaining claims based on a combination of teachings from Zolnowsky and Cameron. Applicants rely on the analysis of the Zolnowsky patent set forth above and maintain that Zolnowsky does not teach or suggest the claimed invention. The Cameron patent is directed to scheduling tasks across multiple nodes (with node defined at Column 2, line 40 as a single processor location) wherein, as specifically stated in Column 2, lines 53-58 and again at Column 7, lines 37-42, "...only one application program is active at a time on any one node and an entire application program is active at once across all of the nodes on which the application program is loaded." In the Cameron system, all actions are initiated from a central dispatcher. While multiple applications can be

assigned to a single processor, only one can be active and ready to run at a time. (Applicants direct the Examiner's attention to the statement in Col. 4, line 5 et seq of Cameron that "although more than one application is assigned in a partition, an entire application is scheduled at once across all the nodes on which it is loaded"). Those applications are assigned by the central dispatcher for the partition. The Cameron global scheduler issues a single directive to execute a task at a particular time and has no capability to prioritize tasks or to dynamically assign tasks of multiple processes in order of importance to utilize idle CPU time. The Cameron patent provides parallel tasks but single level global scheduling with no means for deciding what process or task should execute when a single process of the currently-scheduled parallel job is suspended or waiting. Clearly, therefore, the Cameron patent does not supply the missing teachings to obviate the invention as claimed. Neither Cameron nor Zolnowsky teaches or suggests means or steps for providing application information and local process information to a global scheduler; dynamically creating, updating, and communicating a global prioritized schedule of a plurality of tasks, the schedule including

tasks of more than one application; determining correspondence between the plurality of tasks and the plurality of local processes (which is not the same as Zolnowsky selecting a task from either the global or local queue); and dynamically prioritizing local processes in accordance with the global prioritized schedule to allow simultaneous execution of tasks from more than one application. Accordingly, Applicants respectfully maintain that the claims are patentable over the combination of Zolnowsky and Cameron.

Based on the foregoing amendments and remarks, Applicants respectfully request entry of the amendments, reconsideration of the amended claim language in light of the remarks, withdrawal of the rejections, and allowance of the claims.

Respectfully submitted,

M. A. Cohen, et al

By: Anne Vachon Dougherty

Anne Vachon Dougherty
Registration No. 30,374
Tel. (914) 962-5910